Løsningsantydning til kontinuasjonseksamen i 45060 Systemering 1 Tirsdag 23. august 1994 Kl. 0900 – 1300

•

31. august 1994

Oppgave 1, 15%

Se sidene 346 og 349:

Delsystemstruktur En oppdeling av systemet i en mengde delsystemer, sammen med en mengde korrelasjoner mellom delmengdene. Det er her viktig å få med både delmengdene *og* korrelasjoner.

Konstruktive delsystemstruktur er slik at egenskapene til systemet kan utledes når systemkomponenters egenskaper er kjente.

Anvendbare delsystemstrukturer er slik at egenskapene til systemkomponentene sammen med korrelasjoner mellom dem resulterer i de egenskaper som er definert for totalsystemet.

Implementerbare delsystemstrukturer er slik at hver systemkomponent og korrelasjon kan implementeres.

Fundamentalprinsippet finnes på side 351:

Prinsippet er å inndele systemarbeide i fire separate deloppgaver:

- Definisjon av systemet som en mengde deler. Oppgaven er å liste opp systemkomponentene.
- Definisjon av systemstruktur. Definere alle relevante sammenhenger melom systemkomponenter.

- Definisjon av systemkomponenter. Dvs. for hver av komponentene, definer relevante egenskaper.
- Bestem systemegenskapene. Dette gjøres basert på komponentegenskaper og deres sammenhenger. Resultatet sammenlignes med ønskede systemegenskaper, hvis man ikke er fornøyd repeteres alle fire punkter til man er det.

Oppgave 2, Ytelse (20%)

Se sidene 121-124:

Fixing-point-principle Fikspunktet (dvs. det tidspunktet som binder sammen handlingen med den koden som utfører handlingen, eller det ønskede resultatet med de dataene som produserer det) skal være så tidlig som mulig. Sortering av data kan f.eks. gjøres ved at nye data settes inn i riktig rekkeføge, eller ved å sortere dataene ved behov. I det første tidspunktet har vi et tidlig fikspunkt, mens fikspunktet er sent i det andre tilfellet. Et sent fikspunkt gir større fleksibilitet, men ofte større forbruk av maskinressurser.

Locality-design-principle Lag handlinger, funksjoner og resultater som er mest mulig nær de fysiske dataressursene (prosessorer eller lagringsmedier). Dette vil si at de mulighetene som de fysiske dataressursene tilbyr, skal brukes mest mulig direkte der det er behov for det. Hvis det finnes en spesiell krets for grafiske beregninger bør man f.eks. bruke denne for grafiske operasjoner istedet for å bruke en mer generell prosessor som sikkert er langsommere. En annen anvendelse avprinsippet er å gruppere sammen data som brukes sammen, og prosedyrer som kaller hverandre.

Process-versus-frequency-tradeoff-principle Minimaliser produktet av behandlingstid og frekvens. Prinsippet kan brukes der hvor overheadkostnadene er store. Tenk f.eks. på pakker som sendes over et nett. Ofte vil de få et hode av en fast lenge, slik at det samlede volumet blir mindre hvis flere meldinger grupperes sammen og sendes med kun ett hode.

Shared-resource-principle Del ressurser hvis det er mulig. Dette kan oppnås gjennom å redusere tiden en ressurs er i bruk, eller ved å bruke mindre deler av ressursen. Man kan tenke på låsing av en database ved oppdatering. Det er mulig å låse hele databasen, mens det beste er å kun låse de relasjonene som blir oppdatert. Databasen kan også låses i lengre eller kortere tid. Legg merke til hvordan de siste løsningene er mer komplisert enn de første, "råkraft"-løsningene.

Parallell-processing-principle Kjør forskjellige prosesser i parallell kun hvis dette veier opp for kommunikasjonoverheaden og ressurskonkurransen som dette skaper. Det er mulig å tenke seg eksempler der det å kjøre jobber i parallell gir dårligere responstid enn å kjøre jobbene i serie. Tenk deg at to jobber

skal lese fra en travel disk. Hvis vi kjører oppgavene i parallell, må deres bruk av disken koordineres. Dette slipper vi hvis vi kjører den ene etter den andre.

Centering-principle Identifiser den dominerende lasten på en applikasjon og minimaliser denne. Etter 80-20 regelen vil ofte 20 % av transaksjonene, prosessene eller instruksjonene i et informasjonsystem bruke 80 % av ressursene. Hvis man kan senke ressursforbruket for disse 20 % transaksjonene, prosessene eller instruksjonene litt, er det mye bedre betalt enn å redusere ressursforbruket i de resterende 80 % av transaksjonene, prosessene eller instruksjonene.

Instrumenting-principle Instrumentér koden slik at du kan holde en oversikt over lasten, ressursforbruket og i hvilken grad ytelseskravene er oppfylt. En slik instrumentering bør helst gjøres mens programvaren er under design, fordi mer informasjon er tilgjengelig da enn når kun koden er tilgjengelig.

Oppgave 3, Modellering (50%)

a) (20%) Se vedlagt håndtegning.

En del viktige poeng:

- At det er biltilsynets system som modelleres, i.e. ikke flyt mellom andre aktører.
- At hovedprosessene og aktørene er representert.
- At det finnes datalagre
- At sympoler er brukt korrekt

b) (5%) Timere er kort forklart på side 435, og brukt i et større eksempel i kapittel 13.

Timere er enten klokker eller delays. Klokker brukes til å modellere hendelser som skjer på et gitt tidspunkt, f.eks en spesiell dato, mens delays brukes til å modellere hendelser som skjer på et tidspunkt en gitt tid etter at en annen hendelse har skjedd.

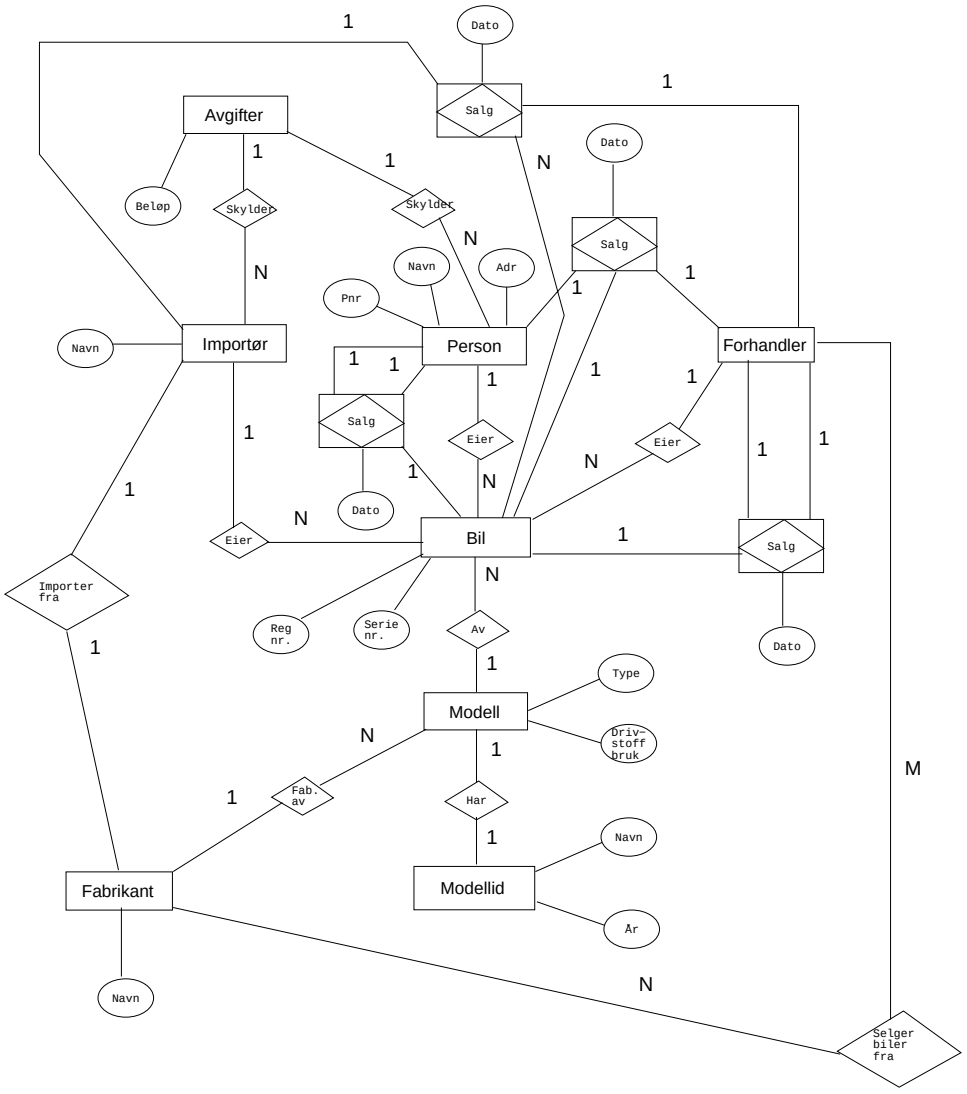
I dette tilfelle kan man bruke en timer som klokke, som 1 gang i måneden sender et tidssignal til prosessen for å purre opp regninger.

c) (10%) Se vedlagt håndtegning.

Et par momenter:

- Sjekk konstruktiv dekomponering
- Forøvrig, se a)

d) (15%) ER-diagram: Merk at det er eksplisitt bedt om ER-diagram, og at ER ikke har generaliseringskonstruksjoner.



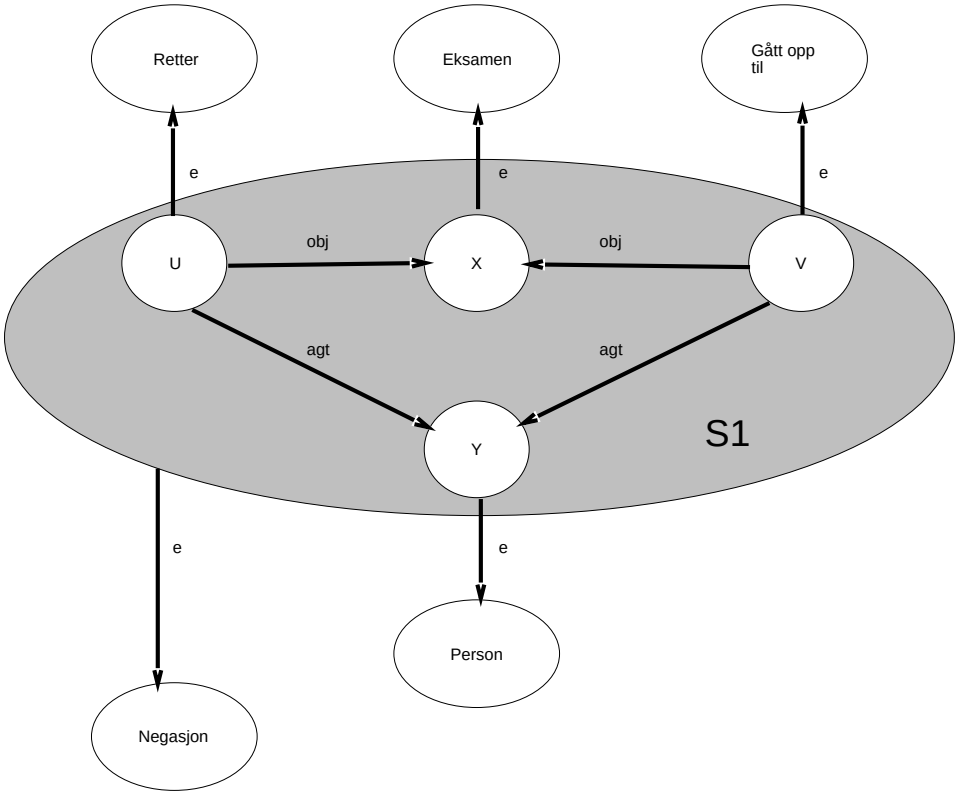
Figur 1: ER-diagram for biltilsynets system

Oppgave 4, Semantiske nett (15%)

Semantiske nett omtales på side 485 til 491

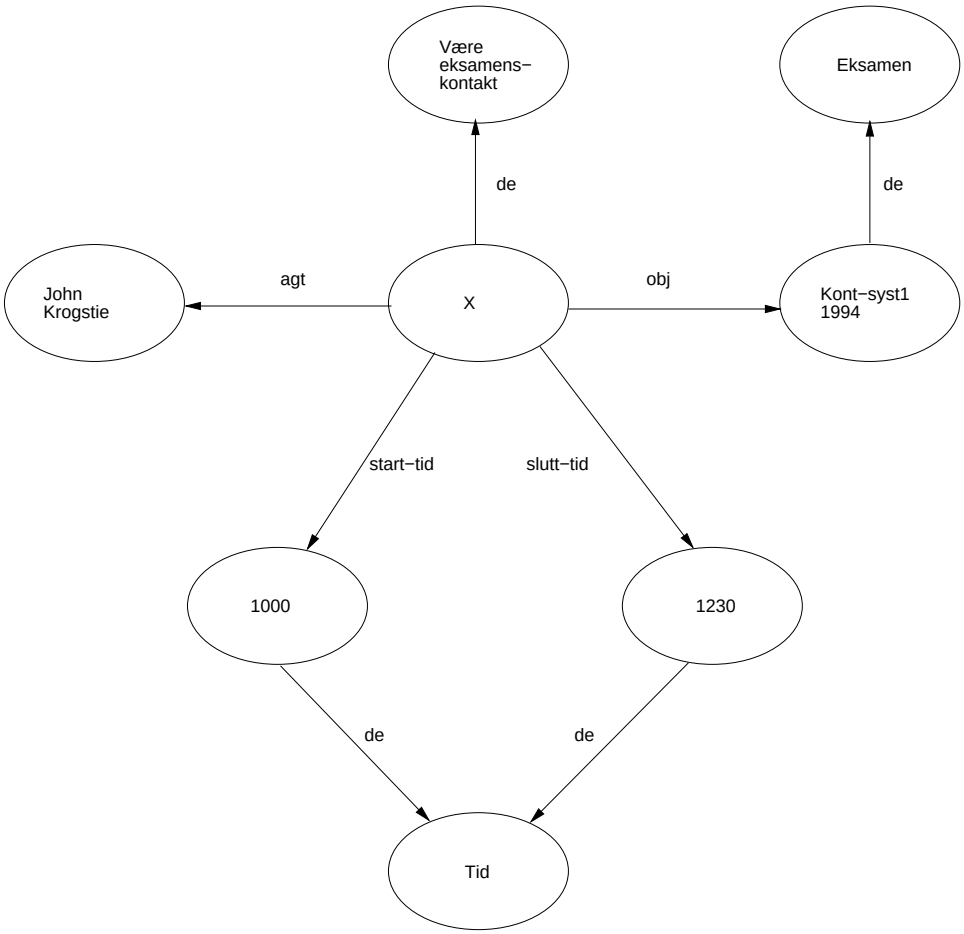
a)

b) Se figur 3



Figur 2: Ingen av de som retter eksamensbesvarelser har gått opp til den samme eksamen

Strengt tatt er det ikke spesifisert i setningen at det er akkurat denne eksamenen det er snakk om, men dette er regnet som implisitt :-)



Figur 3: John Krogstie er kontaktperson under systemeringseksamen mellom 1000 og 1230